

REMARKS

Claims 1, 3, 7, 10, 11, 14-20, and 27-31 are pending in the present application as amended. Claims 6, 8, and 9 have been canceled and the language thereof has been incorporated into independent claim 1, and claim 7 has been amended to depend from claim 1. Likewise, claims 24-26 have been canceled and the language thereof has been incorporated into independent claim 20. Applicants respectfully submit that no new matter has been added to the application by the amendment.

Claims 1, 3, 7, 10, 11, 14-15, 18-20, and 27-29 stand rejected under 35 U.S.C. § 103(a) as being obvious over You (U.S. Patent No. 6,158,045) in view of Niemi et al. (U.S. Patent No. 6,470,388). Applicants respectfully traverse the You-Niemi rejection insofar as it may be applied to the claims as amended.

Independent claim 1 as amended recites a debugger for debugging any of a plurality of debuggees. Each debuggee has a debugging type attribute selected from a plurality of debugging type attributes and representative of a type of debugging to be performed with respect to the debuggee, and each debuggee also has a processor attribute selected from a plurality of processor attributes and representative of a type of processor associated with the debuggee. The debugger is instantiated on a computer and has a single debugger engine for performing debugging functions with respect to any of the plurality of debuggees.

The engine includes a plurality of debugging type blocks, where each debugging type block supports at least one of the plurality of debugging type attributes, and a plurality of processor blocks, where each processor block supports at least one of the plurality of processor attributes. A particular debugging type block and a particular processor block

are selected for debugging a particular debuggee based on the debugging type attribute and processor attribute of the particular debuggee.

Significantly, the plurality of debugging type blocks are organized into a debugging type abstraction available to provide debugging type services that vary in implementation for each debugging type. The debugging type abstraction comprises programming code, and at least a portion of the programming code for the debugging type abstraction is common as between at least some debugging type blocks and is shared by such debugging type blocks. In particular, the programming code for the debugging type abstraction is organized into a tree form with generic code at a base node and more specific levels of code branching out at nodes therefrom. Thus, each debugging type block includes at least one node from the tree.

Likewise, the plurality of processor blocks are organized into a processor abstraction available to provide processor services that vary in implementation for each processor. As with the debugging type abstraction, the processor abstraction comprises programming code, and at least a portion of the programming code for the processor abstraction is common as between at least some processor blocks and is shared by such processor blocks. Also as with the debugging type abstraction, the programming code for the processor abstraction is organized into a tree form with generic code at a base node and more specific levels of code branching out at nodes therefrom. Thus, each processor block including at least one node from the tree.

Independent claim 20 recites subject matter similar to that of claim 1, albeit in the form of a computer with the debugger instantiated thereon.

The You reference discloses a debugger portable to multiple operating systems and hardware platforms. However, Applicants respectfully submit that the You reference does not disclose or suggest that a plurality of debugging type blocks be organized into a debugging type abstraction available to provide debugging type services that vary in implementation for each debugging type and that a plurality of processor blocks be organized into a processor abstraction available to provide processor services that vary in implementation for each processor, as is required by claims 1 and 20. In particular, the You reference does not disclose or suggest that at least a portion of the programming code for the debugging type abstraction is common as between at least some debugging type blocks and is shared by such debugging type blocks and that at least a portion of the programming code for the processor abstraction is common as between at least some processor blocks and is shared by such processor blocks, as is also required by claims 1 and 20. Finally, the You reference does not disclose or suggest that the programming code for the debugging type abstraction is organized into a tree form with generic code at a base node and more specific levels of code branching out at nodes therefrom such that each debugging type block includes at least one node from the tree and that the programming code for the processor abstraction is organized into a tree form with generic code at a base node and more specific levels of code branching out at nodes therefrom such that each processor block includes at least one node from the tree, as is additionally required by claims 1 and 20.

Once again, the Examiner points to Fig. 9 of the You reference as showing such a tree form of an abstraction. However, and again, Applicants respectfully point out that such Fig. 9 in fact shows an addressing abstraction utilized to facilitate the use of target memory addresses in a portable fashion (Abstract), and that such an addressing abstraction is

not at all the debugging type abstraction or the processor abstraction as recited in claims 1 and 20. In particular, the target memory address abstraction contains classes and sub-classes necessary to locate various addresses depending on operating system and/or platform.

In contrast, the debugging type abstraction and the processor abstraction of claims 1 and 20 contain code necessary to define multiple types of debugging type blocks and processors, where such code is organized in a tree form from more generic code to more specific code. Moreover, Applicants respectfully submit that the target memory address abstraction shown in Fig. 9 of the You reference does not even suggest the debugging type abstraction and the processor abstraction, as is recited in claims 1 and 20, especially inasmuch as the You reference contains no suggestion or hint that by providing such a debugging type abstraction and processor abstraction, multiple types of debuggees and processor types can be handled by a single debugger.

In addition, the You reference and the addressing abstraction tree of Fig. 9 thereof do not appreciate that using such tree is not merely a matter of selecting a particular node thereof, but selecting a plurality of such nodes to form a debugging type block or a processor block, as the case may be. That is, the tree of Fig. 9 has a plurality of nodes, each for a particular type of address, where selecting a particular node is necessary for selecting the corresponding type of address. In contrast, the tree recited in claims 1 and 20 has a plurality of nodes where a particular debugging type block or processor block is defined by selecting a plurality of such nodes, as may best be appreciated with reference to Fig. 4 of the present application.

While the Niemi reference may be said to disclose a debugging type abstraction, such Niemi reference like the You reference fails to disclose or suggest that such

a debugging type abstraction is employed and structured in the manner recited in claims 1 and 20, as set forth above, and that a corresponding processor abstraction likewise be employed and structured in the manner recited in claims 1 and 20, as set forth above.

Thus, the combination of the You and Niemi references does not result in the subject matter recited in claims 1 and 20. Accordingly, , Applicants respectfully submit that the You and Niemi references cannot be applied to make obvious claims 1 or 20 or any claims depending therefrom. As a result, Applicants respectfully request reconsideration and withdrawal of the You-Niemi rejection.

Claims 16-17 and 30-31 stand rejected under § 103(a) as being obvious over the You and Niemi references and further in view of Hawley et al. (U.S. Patent No. 5,533,192). Applicants respectfully traverse the You-Niemi-Hawley rejection insofar as it may be applied to the claims as amended.

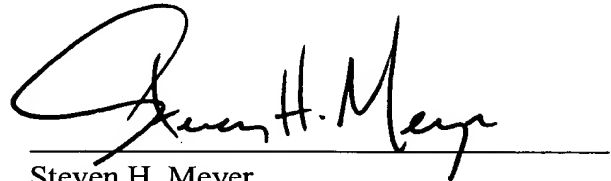
Applicants respectfully point out that since independent claims 1 and 20 are unanticipated and have been shown to be non-obvious, then so too must all claims depending therefrom including claims 16-17 and 30-31 be unanticipated and non-obvious, at least by their dependency. As a result, Applicants respectfully request reconsideration and withdrawal of the You-Niemi-Hawley rejection.

DOCKET NO.: MSFT-0218/146820.01
Application No.: 09/681,064
Office Action Dated: June 14, 2005

PATENT

In view of the foregoing, Applicants respectfully submit that the present application including claims 1, 3, 7, 10, 11, 14-20, and 27-31 is in condition for allowance, and such action is respectfully requested.

Respectfully submitted,

A handwritten signature in black ink, appearing to read "Steven H. Meyer", is written over a horizontal line.

Steven H. Meyer
Registration No. 37,189

Date: September 7, 2005

Woodcock Washburn LLP
One Liberty Place - 46th Floor
Philadelphia PA 19103
Telephone: (215) 568-3100
Facsimile: (215) 568-3439